REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
	Journal Article	

4. TITLE AND SUBTITLE
Time-Parallel Solutions to Differential Equations via Functional Optimization (Pre-Print)

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
C. Lederman, R. Martin, and J-L. Cambier

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
Q02Z

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory (AFMC)
AFRL/RQRS
1 Ara Drive
Edwards AFB, CA 93524

8. PERFORMING ORGANIZATION REPORT NO.

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory (AFMC)
AFRL/RQR
5 Pollux Drive
Edwards AFB, CA 93524

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-RQ-ED-JA-2015-119

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
For publication in Computational and Applied Mathematics; Published March 2016; DOI: 10.1007/s40314-016-0319-7.
PA Case Number: 15222; Clearance Date: 24 April 2015.

14. ABSTRACT

We describe an approach to solving a generic time dependent differential equation (DE) that recasts the problem as a functional optimization one. The techniques employed to solve for the functional minimum, which we relate to the Sobolev Gradient method, allow for large scale parallelization in time, and therefore potential faster "wall-clock" time computing on machines with significant parallel computing capacity. We are able to come up with numerous different discretization and approximations for our optimization derived equations, each of which either puts an existing approach, the Parareal method, in an optimization context, or provides a new time parallel (TP) scheme with potentially faster convergence to the DE solution. We describe how the approach is particularly effective for solving multiscale DEs and present TP schemes that incorporate two different solution scales. Sample results are provided for three differential equations, solved with TP schemes, and we discuss how the choice of TP scheme can have an orders of magnitude effect on the accuracy or convergence rate.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					J. L. Cambier
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NO** *(include area code)*
Unclassified	**Unclassified**	**Unclassified**	SAR	29	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. 239.18

Time-Parallel Solutions to Differential Equations via Functional Optimization

C. Lederman* R. Martin† J-L. Cambier‡

April 9, 2015

Abstract

We describe an approach to solving a generic time dependent differential equation (DE) that recasts the problem as a functional optimization one. The techniques employed to solve for the functional minimum, which we relate to the Sobolev Gradient method, allow for large scale parallelization in time, and therefore potential faster "wall-clock" time computing on machines with significant parallel computing capacity. We are able to come up with numerous different discretization and approximations for our optimization derived equations, each of which either puts an existing approach, the Parareal method, in an optimization context, or provides a new time parallel (TP) scheme with potentially faster convergence to the DE solution. We describe how the approach is particularly effective for solving multiscale DEs and present TP schemes that incorporate two different solution scales. Sample results are provided for three differential equations, solved with TP schemes, and we discuss how the choice of TP scheme can have an orders of magnitude effect on the accuracy or convergence rate.

*corresponding author: carl.lederman.ctr@us.af.mil, ERC Inc., Edwards AFB, CA 93524
†robert.martin.81.ctr@us.af.mil, ERC Inc., Edwards AFB, CA 93524
‡jean_luc.cambier@us.af.mil, Air Force Research Laboratory, Edwards AFB, California 93524

Contents

1 Introduction

Many problems in computational physics are characterized as multiscale, whether spatial, temporal, or involving other dimensions. Efficient integration of the dynamics over the large scales of interest is therefore an issue of key importance for many applications. Numerical schemes developed for this purpose must also satisfy accuracy and stability constraints, and a variety of approaches have been developed to tackle these issues [1, 2]. Some of these make use of the particular structure of the dynamical equations, and have a limited range of applicability. Others are more general, but based on general principles and guidelines [1], and requiring further development for specific applications. Here, we examine a relatively recent approach, the Parareal method, [3], which we refer to as being within a more general class of time parallel (TP) methods, and investigate variations of the procedure. The Parareal method is an iterative error correction between two different scales, coarse and fine, and is closely related to the path optimization problem and the multi-grid scheme applied to the time domain. The multiscale perspective employed by the Parareal method, while not reducing the aggregate computations required, as many other approaches, does allow for greater parallelization, and thus faster real time computing with the appropriate infrastructure. In this paper, we present the Parareal method within an optimization framework and show alternatives that can lead to accelerated convergence over a wider range of scales. We first review the standard Parareal algorithm, and discuss some of the modified approaches that have been developed. Various numerical tests are conducted next, and the results are compared. We offer insights on future directions in the conclusion section.

2 Time Parallel (TP) Review

2.1 Motivation for the TP Approach

The fundamental problem of interest can be described as:

$$\frac{dx}{dt} = f(x), \qquad \text{with} \quad x(0) = x_0 \tag{1}$$

where $x(t) : \mathbb{R} \to \mathbb{R}^D$and $f(x) : \mathbb{R}^D \to \mathbb{R}^D$. Equation (1) describes either an ordinary differential equation (ODE) or a spatially discretized, time-dependent partial differential equation (PDE). The numerical procedure for solving for the solution x involves discretizing the time derivative of x and evaluating f at discrete locations [4]. We can express this in a compact form:

$$x_{t+1} = \mathcal{F}(x_t) \tag{2}$$

where $t \in [0, T]$, $\mathcal{F} : \mathbb{R}^D \to \mathbb{R}^D$ is the discrete propagator, and x_t is a numerical approximation to $x(t)$. Several examples of common propagators are given in the examples of Section 4 below. Numerically solving for a differential equation (DE) with forward propagation results in a perfectly sequential numerical procedure; this is the most natural way to model time-dependent behavior. However, sequentially solving DEs is far from ideal from a modern computational perspective, since the approach does not take advantage of the potential parallelization efficiency of GPUs (Graphics Processing Units) and multi-core CPUs (Central Processing Units). While a solver involving parallel computations in the time domain may be much more complex and require a larger number of operations, the parallel method could potentially result in greatly reduced computational time ("wall-clock"), as a result of the simultaneous execution across may processors. To put it another way, a time parallel (TP) approach consists of creating unnatural and less efficient schemes that, nonetheless, are faster because they better utilize existing computational technology. A basic overview of the challenges involved with such an approach is given in [5].

We will focus exclusively here on TP methods, as opposed to other parallelization algorithms that can be applied within a single time step. There are few good alternatives for solving ODEs beyond TP approaches as the number of computations at a single time step is often relatively small. For PDEs, other appropriate parallelization approaches exist such as the decomposition of the spatial domain or the parallelization of a linear solver [6]within a single time step. Time parallel methods are not always mutually exclusive to these, if there is sufficient computing capacity.

Although the important characteristics of a time parallel implementation can be somewhat problem-specific, there are a few, generally applicable desirable properties required, which can give a general perspective.

1. *Accuracy*: The solution from the parallel algorithm should be nearly identical to that of a well-established serial approach. Improved accuracy is given as a motivation for developing some of the TP schemes presented in Section 3.2 and some general accuracy estimates are given in Section 3.4. A discussion of the accuracy results for some specific problems is given in Section 4.3.
2. *Stability*: The error from a TP method should not become infinitely large when the serial propagators are stable and accurate. We discuss how instability may arise in some cases and how it can be addressed in Section 3.3.
3. *Parallelization Efficiency*: The number of required serial computations should be as small as possible. Additionally, data exchange should be minimized, i.e. a processor should be given as large a block of computational work as possible and should have to send and receive as little data as possible to other processors. Some discussion of this is presented in Section 4.4.

2.2 Existing Time Parallel Approaches

The Parareal method [3] involves both coarse-scale and fine-scale solutions, advanced respectively with a coarse propagator $\mathcal{C}$, and a fine propagator $\mathcal{F}$, which are iteratively updated. In this context, the solution is no longer described using t alone, i.e.:

$$x_t \to x_v^u \tag{3}$$

where u is the iteration number, while $v \in [0, V]$ serves a similar purpose to t at each iteration, but a different symbol is used to indicate that it is not time as it is classically understood. The starting point of the method is the coarse solution, obtained by sequentially advancing the coarse propagator $\mathcal{C}$:

Step 0 [Serial]: $$x_{v+1}^0 = \mathcal{C}(x_v^0) \tag{4}$$

After this initial coarse solution is found a two-step iterative procedure is performed. In the parallel step (1), both $\mathcal{C}$ and the computationally intensive fine propagator $\mathcal{F}$ are run once in parallel for each v. In the serial step (2), the $\mathcal{C}$ is applied sequentially.

Step 1 [Parallel, Iterative]: $$\triangle_{v+1}^{u+1} = \mathcal{F}(x_v^u) - \mathcal{C}(x_v^u) \tag{5}$$

Step 2 [Serial, Iterative]: $$x_{v+1}^{u+1} = \mathcal{C}(x_v^{u+1}) + \triangle_{v+1}^{u+1} \tag{6}$$

The fine propagator can be anything more computationally intensive and accurate than the coarse propagator, but the greatest potential speed-up is achieved when this difference is somewhat large, i.e. for large separation of scales. In this scheme, v is to be considered a 'macroscale' variable associated with a time step dv, with $\mathcal{C}$ applied on this scale. Similarly, w is a 'microscale' variable with time step dw, and $\mathcal{F}$ is a compact way of describing a multi-step integration procedure. The

solution x can be expressed in terms of both a macroscale variable $v \in [0, V]$ and a microscale variable $w \in [0, W]$, where W is number of microsteps *per* macrostep.

$$x_{v,0} \equiv x_v, \quad x_{v,W} \equiv x_{v+1} \tag{7}$$

The fine propagator, $\mathcal{F}$, can be written in such a way to signify the number of times a microscale propagator is applied, W, using the following notation:

$$\mathcal{F}(x_v) \equiv \mathcal{F}_{[0,W]}(x_v) = \mathcal{F}_{[W-1,W]} \circ \mathcal{F}_{[W-2,W-1]} \circ \cdots \circ \mathcal{F}_{[0,1]}(x_{v,0}) \tag{8}$$

The Parareal method has been derived via linear algebra as well as in other mathematical contexts; various modifications to the basic Parareal method have also been proposed [7–12], some of which may be particularly well suited to certain types of DEs. In [11], the authors propose employing the compound-wavelet method for adjusting the coarse solution based on the coarse and fine solutions computed in parallel on each interval. This may be particularly useful when the fine solution is noisy. Additionally, the authors claim that in some circumstances this approach may allow the fine time steps to only be computed on a small portion of the time domain. In [9], a method which works well for certain structural dynamics problems is presented and [10] modifies the original Parareal method to better handle cases where large matrices are involved.

Other types of TP approaches beyond the Parareal method exist as well. The Waveform Relaxation Method [13, 14] solves large systems of differential equations by breaking down the system into loosely coupled subsystems. In [5] an approach to solving linear ODEs in parallel is discussed with a means to extend this approach to a few limited types of ODEs. Lastly, we note that TP approaches have connections to iterative root-finding algorithms, such as Newton approximation methods and Halley's method; that is, the entire DE can be discretized in both space and time, with appropriate boundary conditions, and solved for the unknown variables.

2.3 The Functional Optimization Approach

The problem of finding a solution to a ODE can by reformulated as a problem of finding the minimum of the functional, $\Phi : \mathbb{R}^D \to \mathbb{R}$, given by:

$$\Phi[x] = \frac{1}{2}\int_0^T \left(\frac{d(x(t))}{dt} - f(x(t)\right) \cdot \left(\frac{d(x(t))}{dt} - f(x(t)\right) dt \equiv \frac{1}{2}\int_0^T \varphi(x(t)) \cdot \varphi(x(t)) dt \tag{9}$$

This is an action integral, and the function x that minimizes Φ satisfies:

$$\frac{d(x(t))}{dt} - f(x(t)) \equiv \varphi(x(t)) = 0 \tag{10}$$

At the most basic level, the reason for incorporating this functional is that it quantifies how far away any given function is from satisfying the ODE of interest. Functional optimization techniques can then be employed to find variations of x that can be subtracted to reduce the value of Φ. The following equations are written for a vector-valued $x(t)$ corresponding to an ODE, but generalizes readily to the PDE case with the replacement with the replacement of the discrete dot product:

$$\left(\frac{d(x(t))}{dt} - f(x(t))\right) \cdot \left(\frac{d(x(t))}{dt} - f(x(t))\right) \to \int_\Omega \left(\frac{d(x(z,t))}{dt} - f(x(z,t))\right) \cdot \left(\frac{d(x(z,t))}{dt} - f(x(z,t))\right) dz \tag{11}$$

2.4 The L^2 Gradient

The inner product of the L^2 gradient of the functional and a small variation $h(t)$ is given by:

$$\int_0^T \varphi \cdot \left(\frac{d\varphi}{dx} h\right) dt = \int_0^T \left(\frac{d\varphi}{dx}^T \varphi\right) \cdot h \, dt = \int_0^T (\nabla_{L^2}\Phi(x)) \cdot h \, dt \tag{12}$$

with

$$\nabla_{L^2}\Phi(x) = -\frac{df}{dx}\bigg|_x \left(\frac{d(x(t))}{dt} - f(x(t))\right) - \frac{d}{dt}\left(\frac{d(x(t))}{dt} - f(x(t))\right) \tag{13}$$

This can be computed directly from the functional [15].

Note that here and in what follows, we do not specify boundary conditions. The only boundary condition is at the point $x(0)$ and it remains fixed. One approach to finding the functional minimum is to solve for when this derivative is 0 (the Euler-Lagrange equation) [15].

$$\nabla_{L^2}\Phi(x) = 0 \tag{14}$$

However, this involves solving a large non-linear system, where the size is based on the time step number, and the appropriate nonlinear solver to use is not immediately clear. As an alternative, a popular approach is to employ gradient descent [15]:

$$\frac{dx(u,v)}{du} = -\nabla_{L^2}\Phi(x(u,v)) \tag{15}$$

where x is now considered to be a function of two variables: $x(u,v) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^{\mathbb{D}}$. If implemented explicitly, no linear system needs to be solved:

$$\frac{x_v^{u+1} - x_v^u}{du} = \nabla_{L^2}\Phi(x_v^u) \tag{16}$$

where du is the artificial time step. This approach is, however, impractical for at least two reasons. The first issue is the severe size limitation on du: the L^2 gradient of Φ has a second derivative in v, which yields an oppressive time step restriction of the form $du < K\, dv^2$. The other issue is the speed of propagation of information: if the values of x change near $v \simeq 0$, the values at the end of the time domain, $v \simeq V$, will not be affected until many artificial time steps have occurred.

2.5 The Sobolev Gradient

The Sobolev Gradient can replace the L^2 gradient and lead to a faster numerical computation of a functional minimum [16–24]. The theory behind this approach is well established [16]. One way this has been explained is by comparing the inner products of the standard L^2 and the most common Sobolev inner product, the H^1 inner product [16, 17]. In the L^2 case, it is given by:

$$\langle k, h \rangle_{L^2} = \int_0^T k(t) \cdot h(t) dt \tag{17}$$

This inner product is assumed when computing the L^2 gradient of a functional. The H^1 inner product is given by:

$$\langle k, h \rangle_{H^1} = \int_0^T \left(k(t) \cdot h(t) + \frac{dk(t)}{dt} \cdot \frac{dh(t)}{dt}\right) dt \tag{18}$$

In general, the Sobolev inner product depends only on the two functions and their derivatives. Higher-order derivatives can be included if the functional of interest also contains higher order derivatives [17].

The H^1 gradient is not computed directly from a functional. Instead, the L^2 gradient is computed and then modified based on the relationship between the L^2 and H^1 inner products. With integration by parts,

$$\langle k, h \rangle_{H^1} = \int_0^T \left[\left(I - \frac{d^2}{dt^2} \right) k(t) \right] \cdot h(t)\, dt \tag{19}$$

Taking the directional (Frechet) derivative in the different norms gives [16, 17] :

$$\begin{aligned} \Phi'(x)h &= \langle \nabla_{L^2} \Phi(x), h \rangle_{L^2} \\ \Phi'(x)h &= \langle \nabla_{H^1} \Phi(x), h \rangle_{H^1} \end{aligned} \tag{20}$$

i.e.

$$\int_0^T \left[\left(I - \frac{d^2}{dt^2} \right) \nabla_{H^1} \Phi(x) \right] \cdot h\, dt = \int_0^T \nabla_{L^2} \Phi(x) \cdot h\, dt \tag{21}$$

If this holds for any variation $h(t)$,

$$\nabla_{H^1} \Phi(x) = \left(I - \frac{d^2}{dt^2} \right)^{-1} \nabla_{L^2} \Phi(x) \tag{22}$$

Thus, to obtain the H^1 and the Sobolev gradient in general, the L^2 gradient is computed and a sparse linear system must be solved for. The H^1 gradient is smoother than L^2 and allows much larger time steps to be taken. This stems from the fact the Sobolev inner product also depends on the derivatives of the input functions. This approach has been shown effective in a wide range of applications, including physical modeling [18], image segmentation and registration [19–21], and tetrahedral mesh generation [22].

The Sobolev gradient has been examined specifically in the context of ODEs [23, 24]. While shown to be far better than the L^2 gradient, a few major problems still exist. One issue is that it takes too many iterations to converge; the number of iterations is large enough that it is hard to argue that it is faster than the sequential approach under any reasonable computing system. Additionally, the method does not immediately parallelize well. This is due to the tridiagonal system (22) that must be solved. While these systems can be serially solved quickly [25], less efficient iterative schemes must be employed for parallel solutions [6]. Some of these problems were considered in [24], where it was proposed that the time domain could be broken down into sub-intervals and variable time step sizes employed. But many of the general strategies presented could apply to other optimization approaches and further work is still required to make an approach like this competitive in computation time with the sequential approach.

To make an optimization approach like the Sobolev gradient method viable, we need orders of magnitude reductions in the iteration count as well as a construction that relies only on algorithms that can be done efficiently and directly with parallel computing. We describe such an approach in Section 3 below.

3 Efficient TP Solvers via Functional Optimization

3.1 Derivation of the Time Parallel PDE

While the Sobolev gradient produces a smoother descent that the L^2 gradient, it still does not prevent x from updating to configurations that differ significantly from the true solution of the DE.

Thus, we are led to consider an inner product and an associated norm that more closely resembles Φ:

$$\langle k, h \rangle_{DE}(x) = \int_0^T \left(\frac{dk}{dt} - \frac{df}{dx}\bigg|_x k \right) \cdot \left(\frac{dh}{dt} - \frac{df}{dx}\bigg|_x h \right) dt = \int_0^T \left(\frac{d\varphi}{dx} k \right) \cdot \left(\frac{d\varphi}{dx} h \right) dt \tag{23}$$

with

$$\|h\|_{DE}^2 \equiv \langle h, h \rangle_{DE}(x) \tag{24}$$

The properties of an inner product, including linearity, are satisfied for a given h and k and arbitrary x. Intuitively, this inner product resembles one defined on a manifold, although it is for a function x rather than for a surface. We refer to this inner product as the DE inner product and consider the variations in the solution to be bounded in the DE norm. The variations can be calculated by comparing the L^2 gradient and the DE gradient as is done for the case of the Sobolev gradient.

$$\int_0^T \left[\frac{d\varphi}{dx} \nabla_{DE} \Phi(x) \right] \cdot \frac{d\varphi}{dx} h(t) dt = \int_0^T \varphi \cdot \frac{d\varphi}{dx} h(t) dt \tag{25}$$

If this holds for arbitrary h,

$$\frac{d\varphi}{dx} \nabla_{DE} \Phi(x) = \varphi \tag{26}$$

Combining the gradient descent equation with this gradient yields:

$$\frac{d\varphi}{dx} \frac{dx}{du} + \varphi = 0 \tag{27}$$

Written in terms of $f : \mathbb{R}^D \to \mathbb{R}^D$, this equation is:

$$\frac{d^2 x}{dv du} - \frac{df}{dx}\bigg|_x \frac{dx}{du} + \frac{dx}{dv} - f(x) = 0 \tag{28}$$

with $x(u, v) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^D$. This equation can also be written equivalently using propagators $\mathcal{F}$:

$$\frac{d}{du} x(u, v+dv) - \frac{d\mathcal{F}}{dx}\bigg|_{x(u,v)} \frac{dx(u, v)}{du} + x(u, v+dv) - \mathcal{F}(x(u, v)) = 0 \tag{29}$$

which allows known serial propagators to be encapsulated in this TP PDE more easily.

At first glance, this may seem to be a step backwards, as we have potentially replaced an ODE with a second order non-linear PDE, whose solutions are generally much more computationally intensive. However, this PDE is an artificial one that is designed to solve a computational problem and can be solved significantly faster than an arbitrary PDE of this order. By design, it is well conditioned with step size $du = 1$ and convergences rapidly to the solution of the serial DE in only a few steps in the u direction. Furthermore, unlike the direct serial solution to the DE, solving for this particular PDE can involve parallel processes and can be broken down into two steps: (1) "microscale" and (2) "macroscale". Ten numerical schemes that allow for parallelization are presented in Section 3.2. The first few schemes simply alternate updates in the v and u variables (where v updates use serial DE propagators on independent intervals and u updates combine the independtly computed information) while the rest have a carefully chosen type of u, v stencil that still allows for a TP implementation.

3.2 Numerical Schemes

Scheme 1 (d$\mathcal{F}$): Direct discretization

A first order explicit discretization in the u variable yields:

$$m_v = \frac{x_v^{u+1} - x_v^u}{du} = x_v^{u+1} - x_v^u \tag{30}$$

$$m_{v+1} - \left.\frac{d\mathcal{F}}{dx}\right|_{x_v^u} m_v + x_{v+1}^u - \mathcal{F}(x_v^u) = 0 \tag{31}$$

The propagator $\mathcal{F}$ can be chosen based on an appropriate choice for the serial solver, with $\frac{d\mathcal{F}}{dx}$ the derivative of this propagator with respect to x. While $\mathcal{F}$ is computed over many time steps, only a single vector of size D needs to be exported to the serial solver along with matrix $\frac{d\mathcal{F}}{dx}$. For the examples presented, we take $\mathcal{F} = \mathcal{F}_{[0,W]}$ to be the repeated application of a few common propagators, with a microscale time step size dw, as given in Equation (8). Differentiating this equation yields

$$\left.\frac{d\mathcal{F}_{[0,W]}}{dx}\right|_{x_v^u} = \left.\frac{d\mathcal{F}_{[W-1,W]}}{dx}\right|_{\mathcal{F}_{[0,W-1]}(x_v^u)} \left.\frac{d\mathcal{F}_{[W-1,W-2]}}{dx}\right|_{\mathcal{F}_{[0,W-2]}(x_v^u)} \cdots \left.\frac{d\mathcal{F}_{[1,2]}}{dx}\right|_{\mathcal{F}_{[0,1]}(x_v^u)} \left.\frac{d\mathcal{F}_{[0,1]}}{dx}\right|_{x_v^u} \tag{32}$$

which is the product of W matrices. The resulting algorithm for this discretization is given in Algorithm 1.

Algorithm 1 : TP Implementation of the d$\mathcal{F}$ scheme

Step 0 [Macroscale, Serial]: Solve for the coarse solution and set to x_v^0
for $u = 0$ to $u <$ max iterations do
  Step 1: [Microscale, Parallel]
    1.1 Compute $\mathcal{F}_{[0,w]}(x_v^u)$ for all w by serial propagation
    1.2 Set $\mathcal{F}(x_v^u) = \mathcal{F}_{[0,W]}(x_v^u)$
    1.3 Compute $\left.\frac{d\mathcal{F}_{[0,W]}}{dx}\right|_{x_v^u}$ from Eq. (32)
  Step 2: [Macroscale, Serial]
    2.1 $m_{v+1} = \left.\frac{d\mathcal{F}}{dx}\right|_{x_v^u} m_v - x_{v+1}^u + \mathcal{F}(x_v^u)$
    2.2 $x_v^{u+1} = x_v^u + m_v$
enddo

Scheme 2 (d$\mathcal{C}$) : Direct Discretization with Propagator Derivative Approximation

For this scheme, the fine propagator derivative in Algorithm 1 is replaced with the derivative of a coarse operator.

$$\left.\frac{d\mathcal{F}}{dx}\right|_{x_v^u} m_v \to \left.\frac{d\mathcal{C}}{dx}\right|_{x_v^u} m_v \tag{33}$$

This substitution will reduce the computational cost of the parallel step. Additionally, for larger systems, $d\mathcal{C}/dx$ could reduce the total computation required in the matrix vector multiplication of the macroscale step.

Scheme 3 (Parareal): Approximation with Coarse Propagators

With the following substitution:

$$\left.\frac{d\mathcal{F}}{dx}\right|_{x_v^u} m_v \to \mathcal{C}(x_v^u + m_v) - \mathcal{C}(x_v^u), \tag{34}$$

we have the Parareal method described in Section 2.

Scheme 4 (PA): Propagator Approximation

The function $r : \mathbb{R}^D \times \mathbb{R}^{\mathbb{D}} \to \mathbb{R}^{\mathbb{D}}$ is given by:

$$r(x_{v,w}^u, x_{v,w+1}^u) \equiv x_{v,w+1}^u - \mathcal{F}_1(x_{v,w}^u) \tag{35}$$

Unlike the propagator $\mathcal{F}$, r does not actually propagate a solution forward in time; rather, it more directly measures the error in x. Inserting into the TP PDE, Equation (28), yields:

$$m_{v,w+1} + \left.\frac{dr}{dx}\right|_{x_{v,w+1}^u} m_{v,w} + r(x_{v,w}^u, x_{v,w+1}^u) = 0 \tag{36}$$

We define the function that directly updates the values of m_v as $n_{v,[w,w+1]} : \mathbb{R}^D \to \mathbb{R}^D$:

$$m_{v,w+1} = n_{v,[w,w+1]}(m_{v,w}) \tag{37}$$

where n is written in such a way as to succinctly represent a composition of functions in a multiscale setting:

$$n_{v,[w,w+2]} \equiv n_{v,[w+1,w+2]} \circ n_{v,[w,w+1]} \tag{38}$$

The derivative of n is given by:

$$d_{v,[w,w+1]} = \frac{dn_{v,[w,w+1]}}{dm_{v,w}} = \left.\frac{dr}{dx}\right|_{x_{v,w}^u} \tag{39}$$

d can be computed over an entire macroscale block as a matrix product

$$d_{v,[0,W]} \equiv d_{[v,v+1]} = \frac{dn_{[v,v+1]}}{dm_v} = \left.\frac{dr}{dx}\right|_{x_{v,w}^u} \left.\frac{dr}{dx}\right|_{x_{v,w-1}^u} \cdots \left.\frac{dr}{dx}\right|_{x_{v,0}^u} \tag{40}$$

Unlike Equation (32), this computation does not depend on a serial propagator. We can advance m on the macroscale by the following equation:

$$\begin{aligned} m_{v+1} = n_{[0,v+1]}(m_0) = n_{[0,v+1]}(0) &= d_{[v,v+1]} n_{[0,v]}(0) + n_{[v,v+1]}(0) \\ &= d_{[v,v+1]} m_v + n_{[v,v+1]}(0) \end{aligned} \tag{41}$$

The algorithm for this propagator approximation (PA) scheme is given in Algorithm 2.

Scheme 5 (mP1): First Multi-Propagator Approximation

In this scheme, two TP updates are combined, one which approximates a fine propagator derivative using a coarse propagator derivate, and a second TP update that approximates the DE using this fine propagator derivative approximation. Both of these TP updates, along with additional DE

Algorithm 2 : TP Implementation of the PA scheme

Step 0: [Macroscale, Serial] Solve for the coarse solution and set to x_v^0
for $u = 0$ to $u <$ max iterations do
  Step 1: [Microscale, Parallel]
    1.1 Compute $x_{v,w}^0$for all w by interpolating the coarse solution
    1.2 Compute $r(x_{v,w}^u, x_{v,w+1}^u)$ and $\frac{dr}{dx}|_{x_{v,w}^u}$ for all w
    1.3 Compute $n_{[v,v+1]}(0)$ by serial linear operations as in Eq. (36)
    1.4 Compute $d_{[v,v+1]}$ as a matrix product from Eq. (40)
  Step 2: [Macroscale, Serial]
    2.1 $m_{v+1} = d_{[v,v+1]}m_v + n_{[v,v+1]}(0)$
    2.2 $x_v^{u+1} = x_v^u + m_v$
enddo

coarse propagator computations, are computed in such a way as to still allow for an efficient parallel implementation.

We first consider the matrix equation:

$$\frac{dX}{dt} - \frac{df}{dx}X = 0 \tag{42}$$

where $\frac{df}{dx} : \mathbb{R} \times \mathscr{M} \to \mathscr{M}$ and $X(t) : \mathbb{R} \to \mathscr{M}$. The solution to Equation (42) is given by the serial operation:

$$X_{t+1} = d\mathcal{F}|_{x_t} X_t \tag{43}$$

An approximation to this solution can be found by the same means as an approximation was found to the original equation of interest given by Equation (2). An initial approximation given by a coarse solution

$$X_{v,[0,w]} = d\mathcal{C}_{v,[0,w]} \tag{44}$$

can be updates on the microscale along the lines of Equations (33) and (36). Specifically, the vector valued function x is replaced with the matrix valued function X and dr/dx is replaced by $d\mathcal{C}$. This equation is given by:

$$M1_{v,[w,w+1]} = d\mathcal{C}_{v,[w,w+1]}M1_{v,[0,w]} - X_{v,[0,w+1]} + d\mathcal{F}_{v,[w,w+1]}X_{v,[0,w]} \tag{45}$$

This equation can be solved for by serial linear operations and it is also equal to:

$$M1_{v,[0,W]} = \sum_{i=0}^{W-1} d\mathcal{C}_{v,[i+1,W]} \left(d\mathcal{F}_{v,[i,i+1]} - d\mathcal{C}_{v,[i,i+1]}\right) d\mathcal{C}_{v,[0,i]} \tag{46}$$

The sum in Equation (46) is an $o(W)$ operation and can potentially be computed without significant cost beyond running the fine propagator on the interval of size dv in some cases. However, here, the sum is simply approximated. Using a discrete Simpson's method at the values at $i = 0$, $i = b_1/2$, and $i = b_1$ where $b_1 = W - 1$ and is divisible by 2, this approximation is given by:

$$\begin{aligned} M1_{v,[0,W]} \approx\ & k_0 d\mathcal{C}_{v,[1,b_1+1]} \left(d\mathcal{F}_{v,[0,1]} - d\mathcal{C}_{v,[0,1]}\right) \\ & + k_{b_1/2} d\mathcal{C}_{v,[b_1/2+1,b_1+1]} \left(d\mathcal{F}_{v,[b_1/2,b_1/2+1]} - d\mathcal{C}_{v,[b_1/2,b_1/2+1]}\right) d\mathcal{C}_{v,[0,b_1/2]} \\ & k_{b_1} \left(d\mathcal{F}_{v,[b_1,b_1+1]} - d\mathcal{C}_{v,[b_1,b_1+1]}\right) d\mathcal{C}_{v,[0,b_1]} \end{aligned} \tag{47}$$

where k_0, $k_{b_1/2}$, and k_{b_1} are DE-independent constants of integration. In order to simplify the sum in equation 47, $d\mathcal{C}$ on the requisite intervals is approximated to be either $d\mathcal{C}h$ or $d\mathcal{C}p$ depending on the interval length. That is,

$$d\mathcal{C}_{v,[0,b_1/2]} = d\mathcal{C}_{v,[1,b_1/2+1]} = d\mathcal{C}_{v,[b_1/2,W-1]} = d\mathcal{C}_{v,[b_1/2+1,W]} = d\mathcal{C}h \tag{48}$$

and

$$d\mathcal{C}_{v,[0,1]} = d\mathcal{C}_{v,[b_1/2,b_1/2+1]} = d\mathcal{C}_{v,[W-1,W]} = d\mathcal{C}p \tag{49}$$

The matrix $d\mathcal{C}h$ is computed simply averaging the derivative of coarse operators run on an intervals of size $b_1/2$. The matrix $d\mathcal{C}p$ is approximated by:

$$d\mathcal{C}p = (d\mathcal{C}h)^{\frac{2}{b_1}} = \exp\left(\frac{2}{b_1}\ln(d\mathcal{C}h)\right) \tag{50}$$

computed by a Taylor expansion. The Taylor expansion is truncated based on the number of matrix multiplications required in the computation of $M1$ which is 3 in this case. It is also assumed that $d\mathcal{C}p$ is a close enough approximation to the true value that the commutation error of $d\mathcal{C}p$ and $d\mathcal{C}h$ can be ignored. An improvement upon the the coarse operator $d\mathcal{C}$ is then given by:

$$dmP1_{[v,v+1]} = dmP1_{v,[0,W]} = d\mathcal{C}_{v,[0,w]} + M1_{v,[0,W]} \tag{51}$$

The complete algorithm is summarized in Algorithm 3.

Algorithm 3 : TP Implementation of the mP1 scheme

Step 0: [Macroscale, Serial] Solve for the coarse solution and set to x_v^0 for $u = 0$ to $u <$ max iterations do

Step 1: [Microscale, Parallel]

1.1 Compute $x_{v,w}^0$for all w by interpolating the coarse solution

1.2 Compute $r(x_{v,w}^u, x_{v,w+1}^u)$ and $\frac{dr}{dx}\big|_{x_{v,w}^u}$ for all w

1.3 Compute $n_{[v,v+1]}(0)$ by serial linear operations as in equation 36

1.4 Compute $d\mathcal{C}_{v,[0,b_1/2]}$ and $d\mathcal{C}_{v,[b_1/2+1,W]}$

1.5 $d\mathcal{C}h = \frac{1}{2}\left(d\mathcal{C}_{v,[0,b_1/2]} + d\mathcal{C}_{v,[b_1/2+1,W]}\right)$

1.6 Compute $d\mathcal{C}p$ by Taylor expanding Eq. (50) to order 3.

1.7 $d\mathcal{C}_{v,[0,W]} = d\mathcal{C}h\, d\mathcal{C}p\, d\mathcal{C}h$

1.8 $$M1_{v,[0,W]} = k_0 d\mathcal{C}h\, d\mathcal{C}h\, d\mathcal{F}_{v,[0,1]} + k_{b_1/2} d\mathcal{C}h\, d\mathcal{F}_{v,[b_1/2,b_1/2+1]}\, d\mathcal{C}h + k_{b_1} d\mathcal{F}_{v,[b_1,b_1+1]}\, d\mathcal{C}h\, d\mathcal{C}h - (k_0 + k_{b_1/2} + k_{b_1}) d\mathcal{C}h\, d\mathcal{C}p\, d\mathcal{C}h$$

1.9 $dmP1_{v,[0,W]} = d\mathcal{C}_{v,[0,W]} + M1_{v,[0,W]}$

Step 2: [Macroscale, Serial]

2.1 $m_{v+1} = dmP1_{[v,v+1]} m_v + n_{[v,v+1]}(0)$

2.2 $x_v^{u+1} = x_v^u + m_v$

enddo

Scheme 6 (mP2): Second Multi-Propagator Approximation

A second iteration on $d\mathcal{C}$ such that

$$dmP2_{v,[0,W]} = dC_{v,[0,W]} + M1_{v,[0,W]} + M2_{v,[0,W]} \tag{52}$$

where $M2$ can be computed as:

$$M2_{v,[0,W]} = \sum_{j=1}^{W-1} \sum_{i=0}^{j-1} d\mathcal{C}_{v,[j+1,W]} \cdot \quad \left(d\mathcal{F}_{v,[j,j+1]} - d\mathcal{C}_{v,[j,j+1]}\right) d\mathcal{C}_{v,[i+1,j]} \cdot \left(d\mathcal{F}_{v,[i,i+1]} - d\mathcal{C}_{v,[i,i+1]}\right) d\mathcal{C}_{v,[0,i]} \tag{53}$$

This sum is computed with a discrete Boole's method at the following (i, j) values, where $b_2 = W-2$:

$(0, 1)$	$(0, b_2/4 + 1)$	$(0, b_2/2 + 1)$
$(0, 3b_2/4 + 1)$	$(0, b_2 + 1)$	$(b_2/4, b_2/4 + 1)$
$(b_2/4, b_2/2 + 1)$	$(b_2/4, 3b_2/4 + 1)$	$(b_2/4, b_2 + 1)$
$(b_2/2, b_2/2 + 1)$	$(b_2/2, 3b_2/4 + 1)$	$(b_2/2, b_2 + 1)$
$(3b_2/4, 3b_2/4 + 1)$	$(3b_2/4, b_2 + 1)$	and $(b_2, b_2 + 1)$

Scheme 7 (hA1): First Higher Accuracy Approximation

Many ways of improving the accuracy and stability of PDEs do not directly apply to the TP PDE. The discretization in Scheme 1 does have a forward Euler resemblance, but the propagator $\mathcal{F}$ can be considered to contain arbitrary accuracy in the v variable, and the derivative with respect to u is already well conditioned with step size $du = 1$. The basis for this higher accuracy method is an approximation to the midpoint discretization for the derivative of the propagator with respect to u.

$$\frac{1}{2}\frac{dr}{dx}\bigg|_{x_{v,w}^{u+1}} m_v + \frac{1}{2}\frac{dr}{dx}\bigg|_{x_{v,w}^{u}} m_v + r(x_{v,w}^u, x_{v,w+1}^u) = 0 \tag{54}$$

The approach is similar to the scheme 5 (mP1) except that d^u is the initial approximation rather than the the derivative of a coarse propagator.

$$X_{v,[0,w]} = d^u_{v,[0,w]} \tag{55}$$

This matrix is then updated to approximate d^{u+1},

$$M1_{v,[w,w+1]} = d^u_{v,[w,w+1]} M1_{v,[0,w]} - X_{v,[0,w+1]} + d^{u+1}_{v,[w,w+1]} X_{v,[0,w]} \tag{56}$$

where $M1$ can be directly computed as:

$$M1_{v,[0,W]} = \sum_{i=0}^{W-1} d^u_{v,[i+1,W]} \left(d^{u+1}_{v,[i,i+1]} - d^u_{v,[i,i+1]}\right) d^u_{v,[0,i]} \tag{57}$$

The sum is approximated by evaluating i at 3 points.

$$\begin{aligned} M1_{v,[0,W]} \approx & k_0 d^u_{v,[1,b_1+1]} \left(d^{u+1}_{v,[0,1]} - d^u_{v,[0,1]}\right) \\ & + k_{b_1/2} d^u_{v,[b_1/2+1,b_1+1]} \cdot \quad \left(d^{u+1}_{v,[b_1/2,b_1/2+1]} - d^u_{v,[b_1/2,b_1/2+1]}\right) d^u_{v,[0,b_1/2]} \\ & \qquad k_{b_1} \left(d^{u+1}_{v,[b_1,b_1+1]} - d^u_{v,[b_1,b_1+1]}\right) d^u_{v,[0,b_1]} \end{aligned} \tag{58}$$

The approximation to half of d^u plus half of d^{u+1} is then given by:

$$dhA1_{v,[0,W]} = d^u_{v,[0,W]} + \frac{1}{2} M1_{v,[0,W]} \tag{59}$$

Algorithm 4 : TP Implementation of the hA1 scheme

Step 0: [Macroscale, Serial] Solve for the coarse solution and set to x_v^0

for $u = 0$ to $u <$ max iterations do

Step 1: [Microscale, Parallel]

1.1 Compute $x_{v,w}^0$for all w by interpolating the coarse solution

1.2 Compute $r(x_{v,w}^u, x_{v,w+1}^u)$ and $\frac{dr}{dx}\big|_{x_{v,w}^u}$ for all w

1.3 Compute $n_{[v,v+1]}(0)$ by serial linear operations as in equation 36

1.4 Compute $d_{v,[1,b_1/2]}^u$ and $d_{v,[b_1/2+1,b_1]}^u$ by matrix multiplication

1.5 $d_{v,[0,b_1/2]}^u = d_{v,[1,b_1/2]}^u d_{v,[0,1]}^u$

$d_{v,[1,b_1/2+1]}^u = d_{v,[b_1/2,b_1/2+1]}^u d_{v,[1,b_1/2]}^u$

$d_{v,[b_1/2+1,b_1+1]}^u = d_{v,[b_1+1,b_1]}^u d_{v,[b_1/2+1,b_1]}^u$

$d_{v,[b_1/2,b_1]}^u = d_{v,[b_1/2+1,b_1]}^u d_{v,[b_1/2,b_1/2+1]}^u$

1.6 $d_{v,[1,b_1+1]}^u = d_{v,[b_1/2+1,b_1+1]}^u d_{v,[1,b_1/2+1]}^u$

$d_{v,[0,b_1]}^u = d_{v,[b_1/2,b_1]}^u d_{v,[0,b_1/2]}^u$

1.7 $d_{v,[0,b_1+1]}^u = d_{v,[b_1/2+1,b_1+1]}^u d_{v,[b_1/2,b_1/2+1]}^u d_{v,[0,b_1/2]}^u$

Step 2: [Macroscale-Microparallel]

2.1 Compute $d_{v,[i,i+1]}^{u+1}$ at at $i = 0$, $i = b_1/2$, and $i = b_1$.

2.2 Compute:

$$\begin{aligned} m_{v+1} =& (1 - \frac{k_0}{2} - \frac{k_{b_1/2}}{2} - \frac{k_{b_1}}{2}) d_{[v,v+1]}^u m_v + \frac{k_0}{2} d_{v,[1,b_1+1]}^u d_{v,[0,1]}^{u+1} m_v \\ &+ \frac{k_{b_1/2}}{2} d_{v,[b_1/2+1,b_1+1]}^u d_{v,[b_1/2,b_1/2+1]}^{u+1} d_{v,[0,b_1/2]}^u m_v \\ &+ \frac{k_{b_1}}{2} d_{v,[b_1,b_1+1]}^{u+1} d_{v,[0,b_1]}^u m_v + n_{[v,v+1]}(0) \end{aligned}$$

2.3 $x_v^{u+1} = x_v^u + m_v$

enddo

Unlike Schemes 1, 2, 4, 5, and 6, this scheme requires additional matrix vector multiplications for the macroscale step due to the fact that $d^{u+1}_{v,[i,i+1]}$, computed at three values of i cannot be precomputed in the macro-parallel step. The complete algorithm is summarized in Algorithm 4

Scheme 8 (hA2): Second Higher Accuracy Approximation

A closer approximation to Equation (54) can be obtained by considering a second iteration to the approximation of d^{u+1}. An approximation to the average of d^u and d^{u+1}, $dhA2$, is given by:

$$dhA2_{v,[0,W]} = d^u_{v,[0,W]} + \frac{1}{2}M1_{v,[0,W]} + \frac{1}{2}M2_{v,[0,W]} \tag{60}$$

where

$$M2_{v,[0,W]} = \sum_{j=1}^{W-1}\sum_{i=0}^{j-1} d\mathcal{C}_{v,[j+1,W]} \quad \left(d\mathcal{F}_{v,[j,j+1]} - d\mathcal{C}_{v,[j,j+1]}\right) d\mathcal{C}_{v,[i+1,j]} \left(d\mathcal{F}_{v,[i,i+1]} - d\mathcal{C}_{v,[i,i+1]}\right) d\mathcal{C}_{v,[0,i]} \tag{61}$$

Additionally, a better approximation to $n_{[v,v+1]}$ can be found. $n_{[v,v+1]}$ is computed based only on the values of d^u where as $nhA2_{[v,v+1]}$ incorporates the difference between $dhA1$ and d^u on the microscale.

$$nhA2_{v,[0,W]} = n_{v,[0,W]} + m1_{v,[0,W]} \tag{62}$$

where $m1$ is defined by the equation

$$m1_{v,[w,w+1]} = \left(dhA1_{v,[w,w+1]} - d^u{}_{v,[w,w+1]}\right) m1_{v,[0,w]} + r(x^u_{v,w}, x^u_{v,w+1}) \tag{63}$$

which is equal to:

$$m1_{v,[0,W]} = \sum_{j=1}^{W-1}\sum_{i=0}^{j} d^u_{v,[j+1,W]} \left(d^{u+1}_{v,[j,j+1]} - d^u_{v,[j,j+1]}\right) d^u_{v,[i+1,j]} r(x^u_{v,i}, x^u_{v,i+1}) \tag{64}$$

The TP implementation of this scheme is similar to that of Algorithm 4 but with 15 (i,j) locations evaluated instead of 3 i locations and a matrix multiplied by a vector up to 5 times serially instead of 3.

Scheme 9 (mP1-hA1): First Multi-Propagator and Higher Accuracy Approximation

This scheme is similar to the hA1 scheme, but rather than compute d^uon $[1, b_1/2]$ and $[b_1/2+1, b_1]$, the approximation to d^u given by $dmP1$, as described in Algorithm 3, is employed on each of these two subregions in order to reduce the number of required matrix multiplications.

Scheme 10 (mP2-hA2): Second Multi-Propagator and Higher Accuracy Approximation

For this scheme, $dmP2$ is computed on $[1, b_2/4]$, $[b_2/4+1, b_2/2]$, $[b_2/2+1, 3b_2/4]$, and $[3b_2/4+1, b_2]$ rather than d^u. As with the mP1-hA1 scheme, this is done to reduce computational cost.

3.3 TP Implementation of A Serial Implicit Scheme

The stability of the serial propagators requires that they be bounded in H^1 in the usual sense. Classical stability analysis (Von Neumann stability analysis) assumes linearity, in which case the serial solution is the obtained from the repeated application of $\frac{d\mathcal{F}}{dx}$ and that the magnitude of this operator is less than 1. The stability of the serial schemes carries over to the TP schemes that only depend on the coarse propagator derivative or only depend on the fine propagator derivative.

But instability may occur for the multipropagator schemes 5 and 6 should the difference $\frac{d\mathcal{F}}{dx} - \frac{d\mathcal{C}}{dx}$ become relatively large. Additionally, the propagator derivative will often be written in terms of a matrix inverse when derived from implicit serial schemes. For both of the aforementioned reasons, we describe a fully TP implicit scheme.

The generic form of the propagator in Equation (2) could be considered to encapsulate an implicit serial scheme. However, we instead explicitly delineate the equation as a time reverse propagation:

$$\mathcal{F}(x_{t+1}) - x_t = 0 \tag{65}$$

For scheme 1 (d$\mathcal{F}$), Equation (31) is replaced by:

$$\left.\frac{d\mathcal{F}}{dx}\right|_{x^u_{v+1}} m_{v+1} = \left(m_v - \mathcal{F}(x^u_{v+1}) + x^u_v\right) \tag{66}$$

Similarly, for the PA scheme, Equation (41) is replaced by:

$$d_{[v,v+1]} m_{v+1} = \left(m_v + n_{[v,v+1]}(0)\right) \tag{67}$$

Furthermore, a matrix equation to replace Equation (42) is:

$$X_t = d\mathcal{F}|_{x_{t+1}} X_{t+1} \tag{68}$$

which is the same equation with the direction of time reversed. Therefore, $M1$ is simply

$$M1_{v,[0,W]} = \sum_{i=0}^{W-1} d\mathcal{C}_{v,[0,i]} \left(d\mathcal{F}_{v,[i,i+1]} - !d\mathcal{C}_{v,[i,i+1]}\right) d\mathcal{C}_{v,[i+1,W]} \tag{69}$$

and $M2$ is

$$\begin{aligned} M2_{v,[0,W]} = \sum_{j=1}^{W-1} \sum_{i=0}^{j-1} & d\mathcal{C}_{v,[0,i]} \left(d\mathcal{F}_{v,[i,i+1]} - d\mathcal{C}_{v,[i,i+1]}\right) d\mathcal{C}_{v,[i+1,j]} \\ & \left(d\mathcal{F}_{v,[j,j+1]} - d\mathcal{C}_{v,[j,j+1]}\right) d\mathcal{C}_{v,[j+1,W]} \end{aligned} \tag{70}$$

$dmP1$ and $dmP2$ can be computed by Equations (51) and (52) respectively and can replace $d_{[v,v+1]}$ in Equation (67).

3.4 Accuracy Considerations

For serial propagators, accuracy can usually be analyzed rigorously in the limit as the time step size goes to zero and error accumulates at each serial step [4]. TP methods are fundamentally more challenging, as the convergence of a TP method depends on the global nature of the solution. Some of the challenges of an accurate TP implementation are given in the example problems of Section 4. From the schemes presented here, one can observe how well each incorporates the change in the propagator along the path of the solution. For schemes 2 (d$\mathcal{C}$) and 3 (Parareal), the approximate error is:

$$\left\|\Phi(x^{u+1}_v)\right\| > o\left(\left\|\frac{d^2\mathcal{F}}{dx^2}\right\| \|x^\infty - x^u\|\right) \tag{71}$$

For schemes 1 (d$\mathcal{F}$), 4 (PA), 5 (mP1), and 6 (mP2) we have

$$\left\|\Phi(x_v^{u+1})\right\| = o\left(\left\|\frac{d^2\mathcal{F}}{dx^2}\right\| \|x^\infty - x^u\|\right) \tag{72}$$

And for schemes 7 (hA1), 8 (hA2), 9 (mP1-hA2), and 10 (mP2-hA2)

$$\left\|\Phi(x_v^{u+1})\right\| < o\left(\left\|\frac{d^2\mathcal{F}}{dx^2}\right\| \|x^\infty - x^u\|\right) \tag{73}$$

4 Sample TP Results and Discussion

4.1 Setup

All of the TP methods are associated with a fine propagator, $\mathcal{F}$, computed based on a microscale time step size dw, as well as a coarse propagator, $\mathcal{C}$, computed with time step size dv. The DEs are initialized with coarse propagators for several different schemes. The microscale time step size dw is fixed at a small value while dv is varied to as large a value as possible. The error reduction after the first TP iteration (Figures 2, 6, and 10) as well as number of TP iterations required for convergence (Figures 3, 7, 11) were recorded. The sequential solution with dw sized fine time steps was considered the 'exact' solution, and the non-serial methods were iterated until the error relative to that solution at the last time point was less than one tenth the error from considering the serial solution with $2\,dw$ sized fine time steps in serial. This balance was chosen to give some justification to employing the dw sized time steps but at the same time avoid considering the region of potentially slow convergence that comes about due to double precision floating point operation error in all of the methods.

The DEs were chosen to have parameters with vastly different scales and be somewhat challenging as discussed in Section 4.2. While the coarse time step size, dv, is varied, particular attention is paid to the case when the coarse/fine step ratio is high. The initial coarse solution must still be solved for serially, and therefore, this case where the coarse solution is as coarse as possible offers the opportunity for the highest possible level of parallelization. In addition to error reduction and convergence rates, particular focus on failure is made. Not all of the TP schemes succeed for all of the given coarse solutions. The failure depends both on the robustness of the TP scheme and the global nature of the solution.

4.2 Example DEs

Example 1. Lotka-Volterra

The Lotka-Volterra equation with a quadratic growth term is given by:

$$\begin{aligned} \frac{dx}{dt} &= \alpha x - \beta xy + \gamma x^2 \\ \frac{dy}{dt} &= \eta xy - \sigma y + \chi y^2 \end{aligned} \tag{74}$$

with

$$\begin{aligned} &\alpha = 10,\ \beta = .01,\ \gamma = .0001,\ \eta = .01,\ \sigma = 10,\ \chi = .00002 \\ &x(0) = 1000,\ y(0) = 1000,\ t \in [0, 10] \end{aligned}$$

This model represents the population of two interacting species, a predator and prey. The quadratic term causes a slow acceleration away from the relatively rapidly rotating population sizes (in phase space) and gives the DE a multiscale nature. The serial coarse propagator is forward Euler and the serial fine propagator is the second order Runge-Kutta scheme.

$$\begin{aligned} x_{t+1} = \mathcal{F}(x_t) \rightarrow & x_{t+1} = x_t + dt\, f(x_t) \qquad \text{Coarse} \\ x_{t+1} = \mathcal{F}(x_t) \rightarrow & x_{t+1} = x_t + dt\, f(x_t + \frac{dt}{2} f(x_t)) \qquad \text{Fine} \end{aligned} \tag{75}$$

The coarse step number was varied, resulting in differing amounts of accuracy in the initial coarse solution, and the fine step number was fixed at approximately 2^{20} steps with rounding as necessary for each TP scheme. Figure 1 plots in phase space the difference between this coarsest possible solution and the fine solution. At this level of coarseness the two solutions begin to differ qualitatively due to the coarse solutions inability to accurately track small scale nonlinearities over large times.

Example 2. Pendulum Equation

The nonlinear pendulum equation is given by:

$$\begin{aligned} \frac{dx}{dt} &= y \\ \frac{dy}{dt} &= \frac{-\alpha \sin(\beta\sqrt{x}))}{\sqrt{x}} \end{aligned} \tag{76}$$

with

$$\alpha = 1,\, \beta = 1, \qquad x(0) = 0,\, y(0) = .5,\, t \in [0, 2000]$$

The solutions are nearly circular (like the classic pendulum approximation) but the significant nonlinearity of the DE can lead to large solution error during the course of TP implementation. The serial propagator for both coarse and fine solutions is the Leapfrog scheme

$$x_{t+1} = \mathcal{F}(x_t) \rightarrow \quad \begin{aligned} x_{t+1} &= x_t + dt y_t + \tfrac{dt^2}{2} a(x_t) \\ y_{t+1} &= y_t + \tfrac{dt}{2}(a(x_t) + a(x_{t+1})) \end{aligned} \tag{77}$$

The fine step number is fixed near 2^{20} time steps and the coarse step number is varied. The serial Propagator is symplectic and a high degree of energy conservation is maintained for both the fine and coarsest solution as shown in Figure 5. In contrast to the Lotka-Volterra example, the coarse and fine solutions are qualitatively similar, but this fact alone is insufficient to guarantee that all TP schemes will converge quickly to the fine solution as discussed in Section 4.3.

Example 3. Stiff non-linear equation

An equation that is stiff, nonlinear, and involving multiple scales, to provide a challenging implementation, is given by :

$$\begin{aligned} \frac{dx}{dt} &= -\frac{1}{\gamma}(x + \beta x^3) + y^2 e^{-\beta y} \\ \frac{dy}{dt} &= \frac{1}{\alpha} e^{-x-\beta}(1 + y) + y e^{-\beta y} \end{aligned} \tag{78}$$

with

$$\alpha = 1000,\, \beta = .1,\, \gamma = .001 \qquad x(0) = 1,\, y(0) = 0,\, t \in [0, 10]$$

It is implemented for both fine and coarse solutions with the backward Euler scheme :

$$\mathcal{F}(x_{t+1}) = x_t \rightarrow \quad x_{t+1} - dt\, f(x_{t+1}) = x_t \tag{79}$$

The solution solution has 2^{15} fine time steps. Stiff DEs offer particular challenge in the serial case and additional difficulties in the TP case as discussed in Section 4.3. Figure9 shows phase space plots of the most accurate solution to the DE and the coarsest approximation.

4.3 TP Method Comparison Discussion

4.3.1 Coarse Approximation Scheme Results

Scheme 2 (d$\mathcal{C}$) and scheme 3 (Parareal) performed similarly for the cases where both schemes were able to converge. For both of these schemes, the error of the TP iteration is dominated by the difference between the first derivative of the coarse propagator and the first derivative of the fine propagator. The Parareal method is more robust to the large nonlinearity present in the pendulum DE and is able to converge to the fine solution in cases where the d$\mathcal{C}$ scheme is not able to converge. However, the number of iterations required for convergence becomes quite large as shown in Figure 7. Schemes 2 and 3 require more iterations to converge when compared to other schemes and this distinction in the rate of convergence expands as the coarse to fine step size ratio expands. Additionally, these two schemes do not converge at all when the coarse to fine step size ratio becomes large enough.

4.3.2 Multi-Propogator Scheme Results

Scheme 1 (d$\mathcal{F}$) performed nearly identically to the PA scheme (Scheme 4). Scheme 1 performs non-linear operations on the microscale, where as the PA method relies on linear approximations at the microscale. It does not appear that the microscale non-linear computations are meaningfully incorporated into the TP update to improve its accuracy in the general case.

Schemes 5 (mP1) and 6 (mP2) are reasonable approximations to the PA scheme. This is demonstrated in the error reduction and convergence results of the Lotka-Volterra and pendulum examples. The approximation from the mP1 scheme is better than the dC scheme (scheme 2) but slightly worse than the PA scheme while the more accurate mP2 scheme provides results very close to the PA scheme.

For the stiff DE, the dF and PA schemes fail due to the linear operators becoming un-invertible to machine precision. The mP2 scheme fails for the same reason. Only the mP1 scheme is able to succeed with the coarsest initial solution due to it being more accurate than the dC scheme yet not so accurate as to involve a machine precision un-invertible linear operator.

4.3.3 Higher Accuracy Scheme Results

The higher accuracy schemes 7 (hA1), 8 (hA2), 9 (mP1-hA1) and 10 (mP2-hA2) provide greater error reduction after the first iteration, faster convergence when the initial coarse and fine solutions disagree to a significant degree, and are able to succeeded in cases where other TP methods fail. However, updating the solution in this way does require several additional matrix vector multiplications at each macroscale time step.

4.4 3.4 GPU Computing Results

Six of the TP schemes were implemented using a single CPU and GPU. The microscale/parallel step in Algorithms 1, 2, 3, and 4 was performed on the GPU with each thread given one value of v. The

macroscale/serial step was performed in serial on the CPU. While a higher degree of parallelization is possible for macroscale/serial step in the case of larger systems and more computational capacity (the operations are simply matrix multiplication and vector addition), even this limited parallelization can demonstrate some of the main factors that determine the computing time. As discussed in Section 2.1, the number of serial computations should be limited as much as possible, which, in the context of the schemes described in Section 3.2, means that the macro-step size,V, should be small. For a fixed total fine time step number that is the product of V and W, a small V is obtained by making the coarse to fine step ratio, W, large. Figures 4 and 8 show that the TP iterations are faster as V decreases. The speed up is limited in this case because order W serial computations are performed on each thread. However, this microscale serialization on each macroscale $[v, v+1]$ block can be treated as a subproblem in which another layer of TP iterations can be performed should the computing capacity be available.

The principal difference amongst the TP schemes was that those that produced the least solution error, d$\mathcal{F}$, PA, mP1, and hA1, took roughly two to three times longer. This is mainly due to the fact that on the microscale, the Parareal and d$\mathcal{C}$ schemes require only the calculation of the fine propagator $\mathcal{F}(x_{v,w})$, where as the others require computations of both the fine propagator and its derivative. The hA1 scheme also requires additional macroscale calculations and macroscale data transfers as described in Section 3.2. This results in a higher computational cost when W is small in comparison to the other schemes. Whether the additional time cost of each TP iteration is worthwhile is highly problem dependent but in some cases may result in faster convergence or an orders of magnitude error reduction, as discussed in Section 4.3.

We hope to further examine TP computing speed increases in the context of larger scale ODEs and PDEs in the near future. In particular, the advantages of the mP1 scheme is not revealed in the present analysis because of the small system size. But all of the schemes, except the Parareal, may show additional speed increases when the linear operations in the macroscale step can be parallelized.

5 Applicability Considerations

The sample results provide some indication of the advantages and limitations of the various TP schemes. Many DEs can be solved perfectly well in a reasonable amount of time with established serial methods. Approaches to solving these equations faster with significant parallel computing capacity has emerged as a challenging research area in its own right with practical applications. Other DEs due to their size or multiscale nature, cannot be solved reasonably using classical serial computations.

We hope to apply these TP schemes to more challenging DEs in the future where we would expect a magnification of the TP performance differences observed here as the coarse and fine solutions diverge more drastically and more parallel computing capacity can be applied. The solutions to these more challenging problems will often exhibit a wide range of complicated behavior such as shocks, instabilities, high dimensionality, and a complex multiscale nature. The methodology and results presented here gives some reason for optimism to the prospect of TP methods that can be arbitrarily improved to fit the challenge of at least some of the more problematic PDEs of widespread interest and will be the subject of future work.

Acknowledgments

The authors wish to acknowledge the support of the Air Force Office of Scientific Research (AFOSR), grant No. 12RZ06COR (PM: Dr. F. Fahroo) for this work. We would also like to thank Dr. Justin

Koo of AFRL at Edwards AFB for many fruitful discussions.

References

[1] W. Engquist, "The heterogeneous multi scale methods," *Commun. Math. Sci.*, vol. 1, pp. 87–132, 2003.

[2] C.W. Gear, I.G. Kevrekidis , "Projective methods for stiff differential equations: problems with gaps in their eigenvalue spectrum," *SIAM J. Sci. Comp.*, vol. 24, pp. 109–1110, 2003.

[3] J.-L. Lions, Y. Maday, G. Turinici, "A parareal in time discretization of pde's," *Comptes Rendus de l Academie des Sciences*, vol. Serie I 332, pp. 661–668, 2001.

[4] Iserles Arieh, *A First Course in the Numerical Analysis of Differential Equations.* Cambridge University Press, 2 ed., 2008.

[5] Gear, "Parallel Methods for Ordinary Differential Equations, Vector and Parallel Processors for Scientific Computing-2," technical report, Rome, 1987.

[6] H. S. Stone, "Parallel Tridiagonal Equation Solvers," *ACM Trans. Math. Software (TOMS)*, vol. 1, 1975.

[7] D. Samaddar, D.E. Newman, R. Sanchez, "Parallelization in time of numerical simulations of fully-developed plasma turbulence using the parareal algorithm," *J. Comput. Phys.*, vol. 229, p. 6558, 2010.

[8] Harden, C. and Peterson, J., "Combining the parareal algorithm and reduced order modeling for time dependent partial differential equations," *Harden, C. and Peterson, J. on-line post.*

[9] Farhat, C. and Cortial, J., "A Time-Parallel Implicit Method for Accelerating the Solution of Nonlinear Structural Dynamics Problems," *Int. J. for Num. Methods in Engr.*, pp. 1–25, 2008.

[10] P.Amodio and L.Brugnano, "Parallel solution in time of ODEs: some achievements and perspectives," *Appl. Numer. Math.*, vol. 59, pp. 424–435, 2009.

[11] G. Frantziskonis, K. Muralidharan, P. Deymier, S. Simunovic, P. Nukala, and S. Pannala, "Time-parallel multiscale/multiphysics framework," *J. Comput. Phys.*, vol. 228, pp. 8085 – 8092, 2009.

[12] S. Engblom, "Parallel in Time Simulation of Multiscale Stochastic Chemical Kinetics," *Multiscale Model. Simul.*, vol. 8, pp. 46–68, 2009.

[13] G. Martin, "A Waveform Relaxation Algorithm with Overlapping Splitting for Reaction Diffusion Equations," *Numerical Linear Algebra with Applications*, vol. 1, pp. 1–7, 1993.

[14] J. Sand and K. Burrage, "A Jacobi Waveform Relaxation Method for ODEs," *SIAM J. Sci. Computing.*

[15] C. Fox, *An introduction to the calculus of variations.* Courier Dover Pub., 1987.

[16] J. Neuberger, *Sobolev Gradients and Differential Equations*, vol. Lecture Notes in Mathematics #1670. Springer, 1997.

[17] R. J. Renka, "Constructing fair curves and surfaces with a Sobolev gradient method," Tech. Rep. CAGD 21, 2004.

[18] Majid, A., and Sial, S., "Approximate solutions to Poisson-Boltzmann systems with Sobolev gradients," *J. Comput. Phys.*, vol. 230, pp. 5732–5738, 2011.

[19] Renka, R.J., "Image Segmentation with a Sobolev Gradient Method," *Nonlinear Analysis*, vol. 71, pp. 774–780, 2009.

[20] Lin, T., Dinov, I., Toga, A., Vese., L., "Nonlinear Elasticity Registration and Sobolev Gradients," *Biomedical Image Registration*, vol. 6204, pp. 269–280, 2010.

[21] Lederman, C., Vese, L., and Chien, A., "Registration for 3D Morphological Comparison of Brain Aneurysm Growth," *Adv. Visual Comput.*, vol. 6938, pp. 392–399, 2011.

[22] Lederman, C., Joshi, A., Dinov, I., Vese, L., Toga., A., and Van Horn, J.D., "The Generation of Tetrahedral Mesh Models for Neuroanatomical MRI," vol. 55, pp. 153–164, 2011.

[23] Mahavier, W. T., "Solving boundary value problems numerically using steepest descent in Sobolev spaces," *Missouri J. of Math. Sci.*, vol. 11, pp. 19–32, 2011.

[24] Mujeeb, D., Neuberger, J.W., and Sial, S., "Recursive Form of Sobolev Gradient Method for ODEs on Long Intervals," *Int. J.Computer Math.*, vol. 85, pp. 1727–1740, 2008.

[25] Conte, S.D., and deBoor, C., *Elementary Numerical Analysis.* New York: McGraw-Hill, 1972.

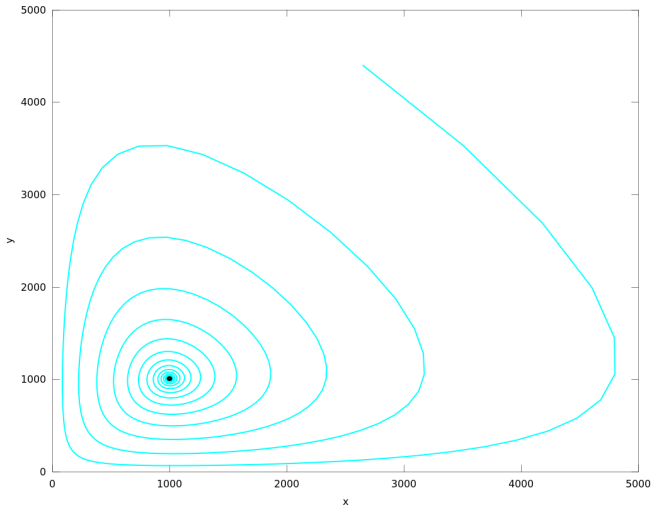

Figure 1: A phase space plot of the Lotka-Volterra example given in Equation 74. The fine solution is black and the coarsest solution provided as an input to each of the TP schemes is in cyan.

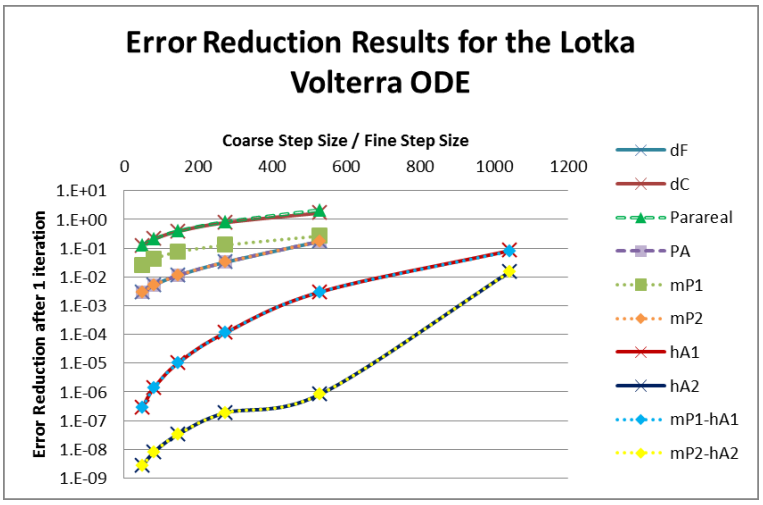

Figure 2: The error reduction after one TP iteration is given for each of the ten schemes presented in Section 3.2 for the Lotka-Volterra example given in Equation 74.

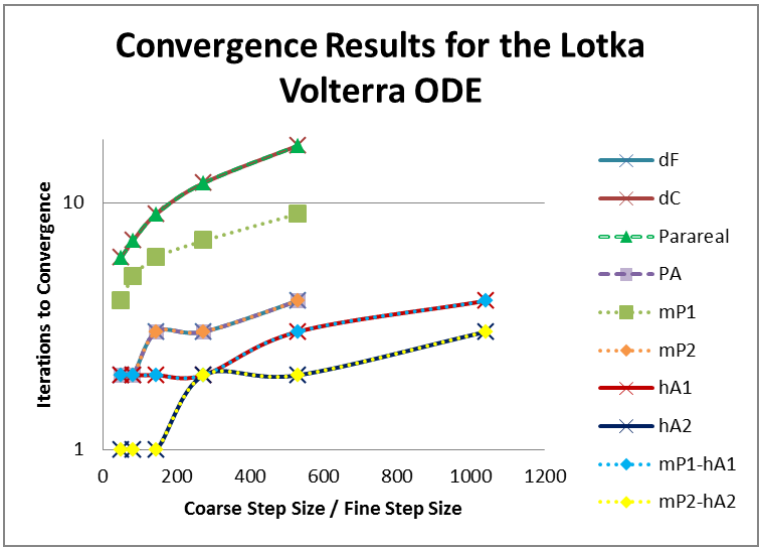

Figure 3: The number of TP iterations until approximate convergence (i.e. when the error is below one tenth the error of a run with twice the step size) is shown for each of the ten schemes presented in Section 3.2 for the Lotka-Volterra example given in Equation 74.

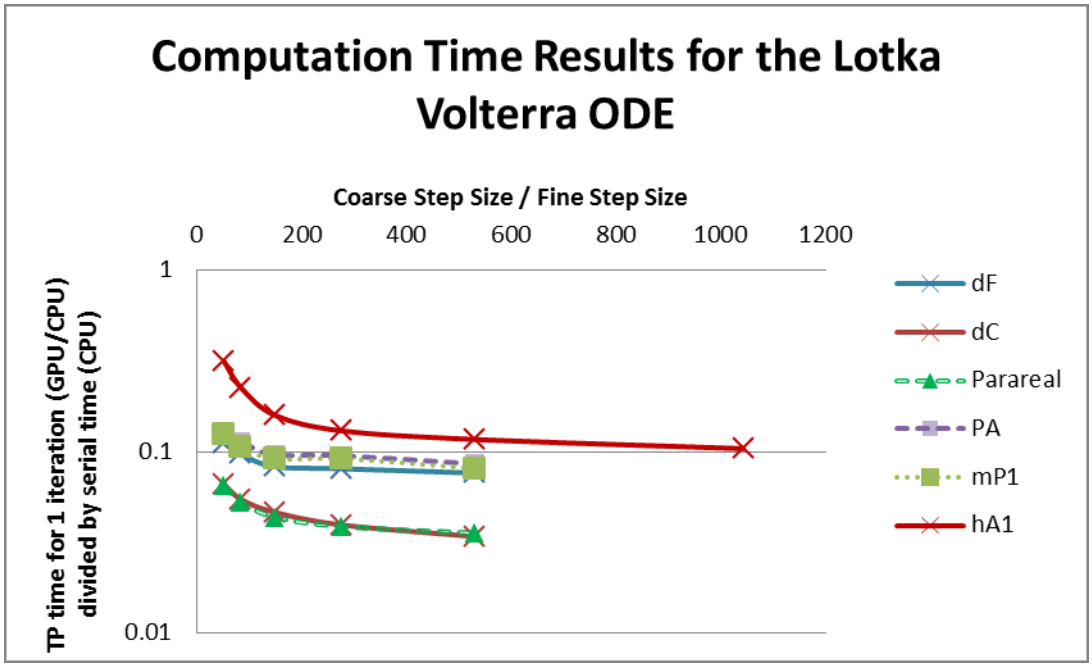

Figure 4: The computation time ("wall-clock") for a single TP iteration was computed for six different TP schemes for the Lotka-Volterra example. The parallel computations were performed on a single GPU and the serial compuations were performed on a single CPU. In general, TP iterations that generated less solution error required modestly higher computation cost.

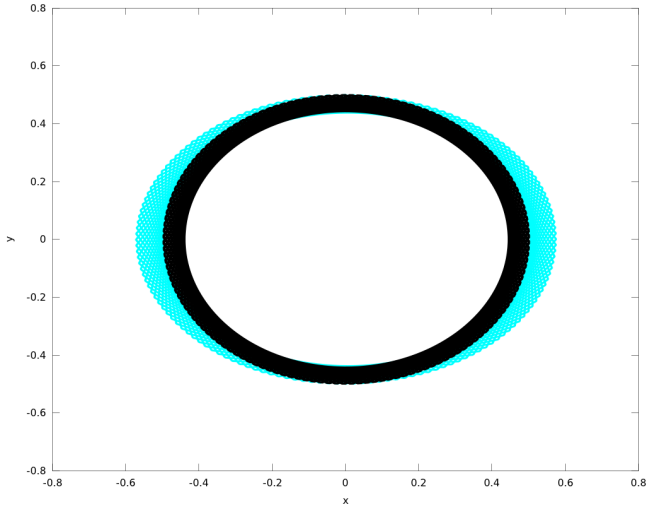

Figure 5: A phase space plot of the pendulum example given in Equation 76. The fine solution is black and the coarsest solution provided as an input to each of the TP schemes is in cyan.

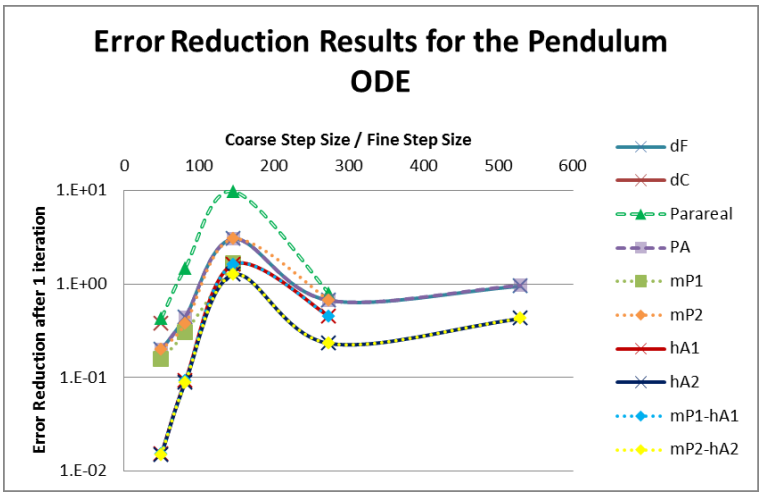

Figure 6: The error reduction after one TP iteration is given for each of the ten schemes presented in Section 3.2 for the pendulum example given in Equation 76.

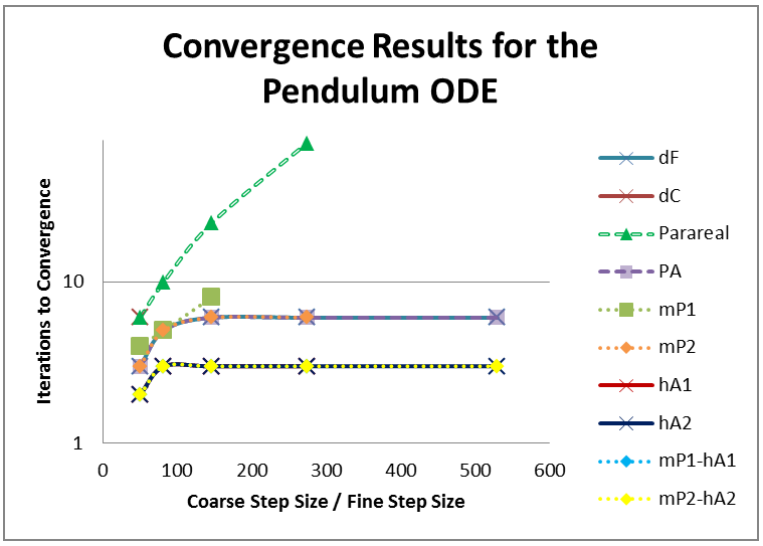

Figure 7: The number of TP iterations until approximate convergence (i.e. when the error is below one tenth the error of a run with twice the step size) is shown for each of the ten schemes presented in Section 3.2 for the pendulum example given in Equation 76.

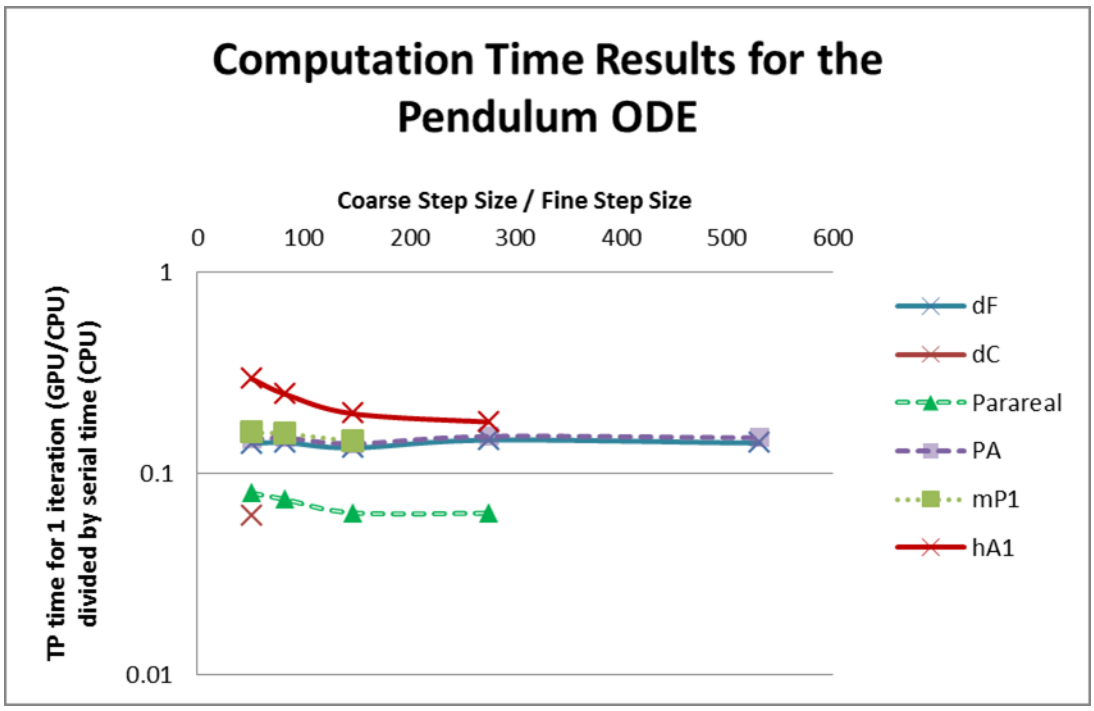

Figure 8: The computation time ("wall-clock") for a single TP iteration was computed for six different TP schemes for the pendulum example. The parallel computations were performed on a single GPU and the serial compuations were performed on a single CPU. In general, TP iterations that generated less solution error required modestly higher computation cost.

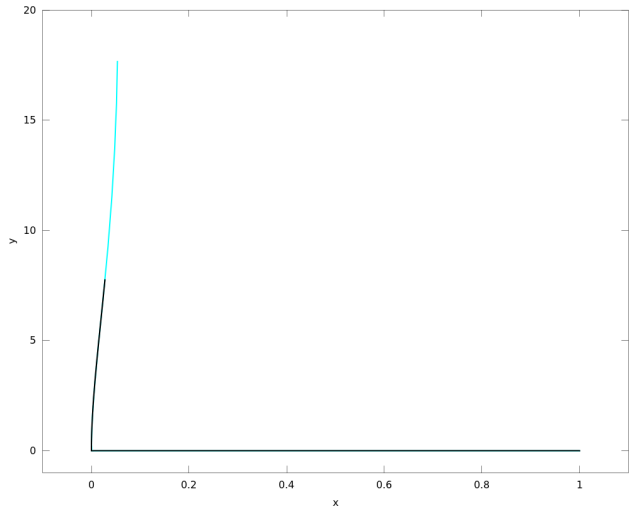

Figure 9: A phase space plot of the stiff example given in Equation 78. The fine solution is black and the coarsest solution provided as an input to each of the TP schemes is in cyan.

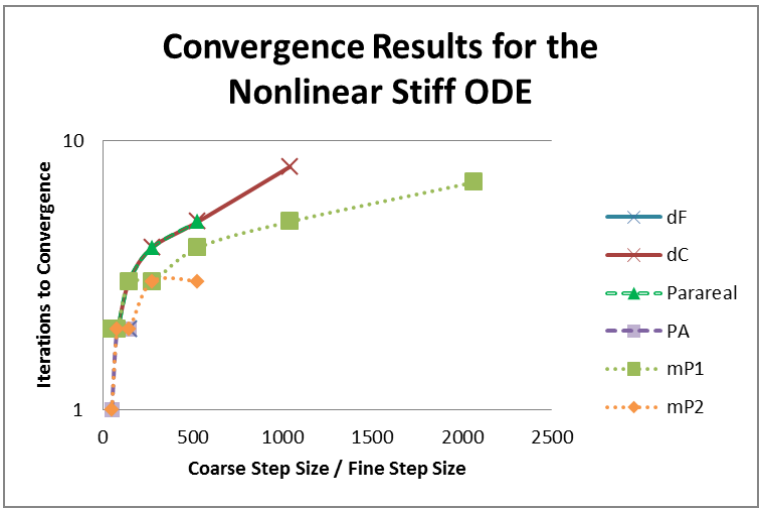

Figure 10: The error reduction after one TP iteration is given for each of the ten schemes presented in Section 3.2 for the stiff example given in Equation 78.

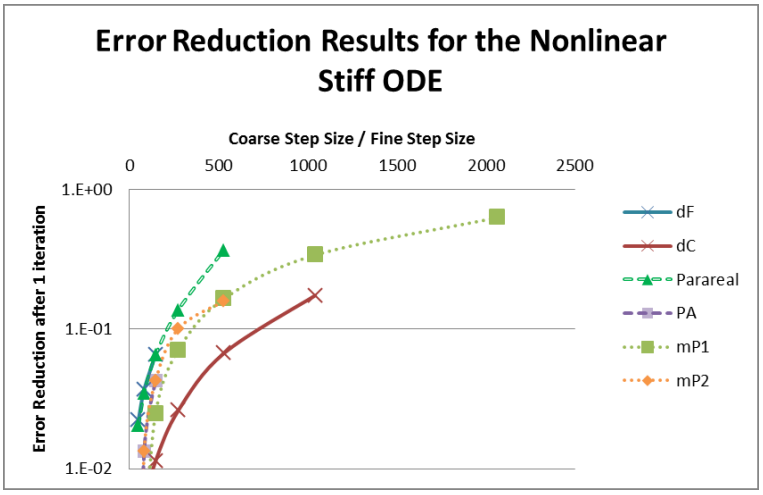

Figure 11: The number of TP iterations until approximate convergence (i.e. when the error is below one tenth the error of a run with twice the step size) is shown for each of the ten schemes presented in Section 3.2 for the stiff example given in Equation 78.